PROBABILISTIC PASSWORD CRACKING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation-in-part of and claims priority to U.S. Nonprovisional patent application Ser. No. 13/624,249, entitled "Password Cracking Through Learning Probabilistic CFGs", filed on Sep. 21, 2012, which is a continuation of and claims priority to U.S. Nonprovisional patent application Ser. No. 13/547,779, entitled "Password Cracking Through Learning Probabilistic CFGs", filed on Jul. 12, 2012, which claims priority to U.S. Provisional Application No. 61/506,785, entitled "Password Cracking Through Learning Probabilistic CFGs", filed on Jul. 12, 2011.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

This invention was made with Government support under Grant No. 2006-DN-BX-K007 awarded by the U.S. National Institute of Justice. The government has certain rights in the invention.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates, generally, to cryptography. More particularly, it relates to a system and method of probabilistic password cracking.

2. Brief Description of the Prior Art

Human memorable passwords provide the basis for much of today's encryption and authentication protocols. This is due to numerous features that passwords possess, such as the lack of additional hardware requirements (e.g., scanners or public/private key tokens), user acceptance, and the ease with which passwords can be transformed into encryption keys via hashing.

In the setting of cracking passwords in a forensic setting, the attacker (e.g., law enforcement) has obtained the password hashes or encrypted files and is now attempting to decrypt the files by figuring out the original passwords from the hashes. A forensic, or offline password cracking attack can be broken up into three distinct steps. First, the attacker makes a guess as to the user's password, for example "password123". Next, the attacker hashes that guess using whatever hashing algorithm was used. In the case of file encryption, the hashing algorithm is used to convert the password guess into an encryption key. Thirdly, the attacker compares the hash of the password guess to the hash the attacker is trying to crack. If the two hashes match, the password is considered broken. With file encryption, the attacker attempts to decrypt the file (or file header) with the key generated, and if the file is decrypted successfully, the password is considered cracked. These three steps are repeated over and over again with new guesses until the attacker breaks the password, or runs out of time. However, this process is very time-consuming (i.e., time that law enforcement might not have) and inefficient in making password guesses.

The two most commonly used methods to make password guesses are brute-force and dictionary based attacks. With brute-force, the attacker attempts to try all possible password combinations. While this attack is guaranteed to recover the password if the attacker manages to brute-force the entire password space, exhaustive search of the password space is often not feasible due to time and equipment constraints. Several techniques have been developed to generate more targeted search spaces, for example Markov models (L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE, V.* 77, No. 2 (February 1989)), and can be used to generate search spaces according to heuristics about the structure of likely passwords. This strategy has been indeed adopted by popular password crackers such as JOHN THE RIPPER™.

If no salting is used, brute-force attacks can be dramatically improved through the use of pre-computation and powerful time-memory trade-off techniques known as rainbow tables (N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, "Time-Memory Trade-Off Attack on FPGA Platforms: UNIX Password Cracking," *Proceedings of the International Workshop on Reconfigurable Computing: Architectures and Applications*, Lecture Notes in Computer Science, V. 3985, pg. 323-334, Springer (2006); M. Hellman, "A Cryptanalytic Time-Memory Trade-Off," *IEEE Transactions on information Theory*, V. 26, Issue 4, pg. 401-406 (1980); P. Oechslin, "Making a Faster Cryptanalytic Time-Memory Trade-Off," *Proceedings of Advances in Cryptology* (*CRYPTO* 2003), Lecture Notes in Computer Science, V. 2729, pg. 617-630, Springer (2003)). Some Markov models may be de-randomized into a deterministic index function, allowing them to be combined with time-memory trade-off techniques, such as the construction of optimized rainbow tables (A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff," CCS '05 (Alexandria, Va. Nov. 7-11, 2005)).

The second main technique of making password guesses is a dictionary attack. The dictionary itself may be a collection of word lists that are believed to be common sources from which users choose mnemonic passwords. However, users rarely select unmodified elements from such lists, for instance because password creation policies prevent it, and instead generally modify the words in such a way that they can still recall them easily. In a dictionary attack, the attacker tries to reproduce this approach to password choice by processing words from an input dictionary and systematically producing variants through the application of preselected mangling rules. For example, a word-mangling rule that adds the number "9" at the end of a dictionary word would create the guess "password9" from the dictionary word "password."

For a dictionary attack to be successful, it requires the original word to be in the attacker's input dictionary and for the attacker to use the correct word-mangling rule. While dictionary based attack is often faster than brute-force on average, attackers are still limited by the amount of word-mangling rules they can take advantage of due to time constraints. Such constraints become more acute as the sizes of the input dictionaries grow. In this case, it becomes important to select rules that provide a high degree of success while limiting the number of guesses required per dictionary word.

Choosing the right word-mangling rules is crucial as the application of each rule results in a large number of guesses. This is especially true when the rules are used in combination. For example, adding a specific two-digit number to the end of a dictionary word for a dictionary size of 800,000 words would result in 80,000,000 guesses. Creating a rule to allow the first letter to be uppercase or lowercase would double this figure. Furthermore, in a typical password retrieval attempt, it is necessary to try many different man-